Exercise 3 | TKO_2096 Applications of Data Analysis 2021

deadline: 22.2.2021, 23:59 pm

Student name: Rami Ilo

Water permeability prediction in forestry

In this task, the client wants you to estimate the spatial prediction performance of K-nearest neighbor regression model with K=7 (7NN), using spatial leave-one-out cross-validation (i.e. SKCV, with number of folds == number of data points). The client wants you to use the C-index as the performance measure.

In other words, the client wants you to answer the question: "What happens to the prediction performance of water permeability using 7-nearest neighbor regression model, when the geographical distance between known data and unknown data increases?".

In this task, you have three data files available (with 1691 data points):

- input.csv, contains the 75 predictor features.
- output.csv, contains the water permebility values.
- coordinates.csv, contains the corresponding geographical coordinate locations of the data points. The unit of the coordinates is metre, and you can use Euclidean distance to calculate distances between the coordinate points.

Implement the following tasks to complete this exercise:

1. Z-score standardize the predictor features (input.csv).
2. Perform spatial leave-one-out cross-validation with 7NN model for the provided data set (refer to the lectures 3.1.3 and 3.1.4 for help). Estimate the water permeability prediction performance (using 7NN model and C-index) with the following distance parameter values: d = 0, 10, 20, ..., 200 (that is, 10 meter intervals from 0m to 200m).
3. When you have calculated the C-index performance measure for each value of d, visualize the results with the C-index (y-axis) as a function of d (x-axis).

Your .ipynb-file must include the following:

- Your own implementation of the spatial leave-one-out cross-validation for the current task. Remember to also take advantage of earlier exercises (e.g. C-index). For the 7-nearest neighbor and Euclidean distance calculation you can use third-party libraries (e.g. Scikit-learn) if you want.
- Plot of the graph C-index vs. distance parameter value.

-- START IMPLEMENTING YOUR EXERCISE AFTER THIS LINE --

Import necessary libraries

In [106]:
```
# In this cell, import all the libraries that you need. For example:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import neighbors
import sklearn
```

Read in the datasets

In [107]:
```
# In this cell, read the files input.csv, output.csv and coordinates.csv.
# Print out the dataset dimesions (i.e. number of rows and columns).
#
# Note that the coordinates are in EUREF-TM35FIN format, so you
# can use the Euclidean distance to calculate the distance between two coordinate points.

X = pd.read_csv(r"G:\Dropbox\Applications of data analysis\ex3\input.csv")
```

```
y = pd.read_csv(r"G:\Dropbox\Applications of data analysis\ex3\output.csv")
coordinates = pd.read_csv(r"G:\Dropbox\Applications of data analysis\ex3\coordinates.csv")

def print_dataset_info(df):
    print(df.shape)

print_dataset_info(X)
print_dataset_info(y)
print_dataset_info(coordinates)

coordinates.head(5)
```

```
(1690, 75)
(1690, 1)
(1690, 2)
```

	4.5914e+05	7.5242e+06
0	461590.0	7549000.0
1	462040.0	7549300.0
2	462040.0	7549300.0
3	462130.0	7549400.0
4	462200.0	7547400.0

Standardization of the predictor features (input.csv)

In [108]:
```
# Standardize the predictor features (input.csv) by removing the mean and scaling to unit variance.
# In other words, z-score the predictor features. You are allowed to use third-party libraries for c

#A function to calculate z-score for a column
def get_z_score(column):
    mean = np.mean(column)
    std = np.std(column)
    z_score = (column - mean) / (std)
    return z_score

print(coordinates.iloc[0])
from scipy.spatial import distance

print(np.linalg.norm(coordinates.iloc[2] - coordinates.iloc[0]))

X = get_z_score(X)
X.head(5)
```

```
4.5914e+05     461590.0
7.5242e+06    7549000.0
Name: 0, dtype: float64
540.8326913195984
```

	-5.319627000693968877e-02	-2.192960385319173422e-01	2.100203710608411767e-01	7.044248857837395184e-01	3.394767803718
0	-0.631943	-0.682742	-0.285317	-0.369070	-0.138068
1	-0.674812	-0.597383	-0.568404	-0.982674	-0.828018
2	-0.331860	-0.154735	1.307047	0.091133	0.419198
3	-0.610509	-0.672986	-0.161466	-0.522471	-0.376897
4	-0.256840	-0.220583	1.360126	0.858138	0.684563

5 rows × 75 columns

Functions

In [109]:
```
# Include here all the functions (for example the C-index-function) that you need in order to implen

#Concordance index function
def cindex(true_labels, pred_labels):
    """Returns C-index between true labels and predicted labels"""

    counter = 0
```

```
    pairs = 0

    for i in range(0, len(true_labels)):
        assert len(true_labels) == len(pred_labels)
        #test every label against all other labels that are larger
        x = true_labels[i]
        for j in range(0, len(true_labels)):
            if(true_labels[j] > x):
                pairs+=1
                if(pred_labels[i] < pred_labels[j]):
                    counter+=1
                elif(pred_labels[i] == pred_labels[j]):
                    counter+=0.5

    cindx = counter / pairs
    return cindx
```

Results for spatial leave-one-out cross-validation with 7-nearest neighbor regression model

In [148]:

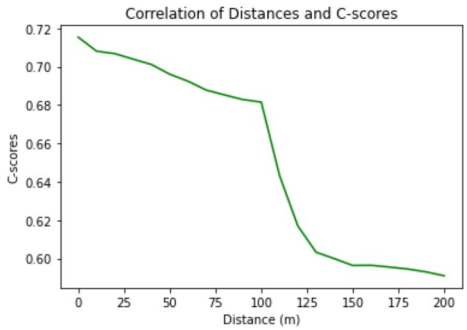
```
# In this cell, run your script for the Spatial leave-One-Out cross-validation
# with 7-nearest neighbor regression model and visualize the results as
# requested in the task assignment.

#  d = 0, 10, 20, ..., 200 (that is, 10 meter intervals from 0m to 200m)
distance = [(lambda d: d)(d) for d in range(0, 210, 10)]
n_neighbors = 7

def get_distance_indices(h, i, d):
    a = np.array(np.where(h<=d))
    k = 0

    for j in range(0, i):
        if (h[j]<=d):
            k+=1

    return a, k

"""As the number of folds = N, leave-one-out cv
Basing the spatial part of the function on the pseudocode on the lecture slide 3.1.4. page 2
"""
def sloocv(X, y, c, d, n_neighbors, title):
    scores = []
    true_labels = []
    #Get the matrix of pairwise distances
    e = neighbors.DistanceMetric.get_metric('euclidean')
    h = e.pairwise(c)

    #A fold with one sample for test and the rest for training
    for i in range(0, len(y)):
        #Remove data points too close. Respect to the point i, find the list of points that are too
        #according to the coordinates list c
        dist_less_than, i_reduce = get_distance_indices(h[i], i, d)

        dist_less_than = np.append(dist_less_than, i)

        #Remove the too close points
        XD = np.array(np.delete(X, dist_less_than, axis=0))
        yD = np.array(np.delete(y, dist_less_than, axis=0))

        #Split the data: in every loop, everything but i:th instance is chosen as training data minu
        #the points too close, which were removed above.
        (train_data, val_data, train_labels, val_labels) = XD, X[i], yD, y[i]
```

```
            knn = neighbors.KNeighborsRegressor(n_neighbors)
            knn.fit(train_data, train_labels)
            #Predict on the i:th element. 2D array is required for predict(), so arbitarily reshape to 1
            val_pred = knn.predict([val_data])

            #Collect the predictions and correct labels in their respective arrays
            scores = np.append(scores, np.array(val_pred))
            true_labels = np.append(true_labels, np.array(val_labels))

        scores = np.array(scores)
        true_labels = np.array(true_labels)
        #Get the score from the 1...N predictions
        c_score = cindex(true_labels, scores)
        print("C-index score for {1} with distance of {2} meters: {0}]".format(c_score, title, d))
        return c_score

    c_scores = []
    for di in distance:
        c_scores.append(sloocv(X.to_numpy(), y.to_numpy(), coordinates, di, n_neighbors, "Water permeabi
```

```
C-index score for Water permeability with distance of 0 meters: 0.715296348621693]
C-index score for Water permeability with distance of 10 meters: 0.7080123730875535]
C-index score for Water permeability with distance of 20 meters: 0.7067197722723986]
C-index score for Water permeability with distance of 30 meters: 0.7038745788272758]
C-index score for Water permeability with distance of 40 meters: 0.7010707317893604]
C-index score for Water permeability with distance of 50 meters: 0.6960050961198849]
C-index score for Water permeability with distance of 60 meters: 0.6923494430148398]
C-index score for Water permeability with distance of 70 meters: 0.6877039288897859]
C-index score for Water permeability with distance of 80 meters: 0.6852003688940467]
C-index score for Water permeability with distance of 90 meters: 0.6827973717700747]
C-index score for Water permeability with distance of 100 meters: 0.6814735859528736]
C-index score for Water permeability with distance of 110 meters: 0.6431748995072069]
C-index score for Water permeability with distance of 120 meters: 0.6168933010410885]
C-index score for Water permeability with distance of 130 meters: 0.6032083410419855]
C-index score for Water permeability with distance of 140 meters: 0.5998571796984936]
C-index score for Water permeability with distance of 150 meters: 0.5963322232874178]
C-index score for Water permeability with distance of 160 meters: 0.5964184200346468]
C-index score for Water permeability with distance of 170 meters: 0.5954954841313891]
C-index score for Water permeability with distance of 180 meters: 0.594493709739812]
C-index score for Water permeability with distance of 190 meters: 0.5929901721692428]
C-index score for Water permeability with distance of 200 meters: 0.5909536864176351]
```

Interpretation of the results

In [149]:
```
#Display plot
plt.plot(distance, c_scores, color='green')
plt.xlabel('Distance (m)')
plt.ylabel('C-scores')
plt.title("Correlation of Distances and C-scores")

plt.show()
```



In [ ]:
```
# In this cell, give a brief commentary on the results, what happens to the prediction
# performance as the prediction distance increases?
"""According to this notebook, the score is reversely related to the distance. This kind of pattern
can also be expected, as "objects geographically closer to each other are more similar", and as the
distance parameter increases, the training and test set are going to get samples that would,
according to abovementioned rule, be less and less like each other, i.e. making the task harder.
```

```
On the other hand, bias must also be taken into consideration, as only one score is reported per dis
and multiple close points might increase the average score."""
```